UE055.1 – Programmation objet

Fiche TP2

Les figures

Environnement de travail

Vous pouvez développer vos classes à l'aide de n'importe quel éditeur de texte et compiler et exécuter vos programmes à l'aide des commandes suivantes

1. Compilation : javac maClasse.java
2. Exécution : java maClasse

Vous pouvez également utiliser un environnement de développement (Conseillé). Si vous utilisez *Eclipse* (conseillé), commencez par définir l'emplacement de votre espace de travail (*workspace*) puis créez un nouveau projet soit vide soit à partir de classes java déjà existantes. Vous pourrez ensuite compiler et exécuter directement vos programmes depuis l'environnement *Eclipse*.

Travail à faire

Pour cette première série de TPs vous devez coder l'ensemble des figures (***Point, Cercle, Segment, Polygone, PointNomme, …***) vues en TD. Ces classes seront mises dans le paquetage ***Figures***
Pour chaque classe vous devez **coder les méthodes suivantes**

- ***le constructeur,***
- ***public String toString() :*** conversion des attributs de la classe en chaîne de caractères.
- ***public void afficher()*** : affichage des attributs de la classe.
- ***public void translater(double dx, double dy)*** : translation de la figure
- ***public Point getCentre()*** : renvoi du centre de la figure.

Par la suite vous aurez d'autres méthodes à ajouter à ces classes.
Vous devez également coder un **programme de test** qui comportera une méthode **main()** qui fera appel à des méthodes que vous coderez au fur et à mesure en fonction de l'avancement des TPs. Par exemple après le codage de la classe Point vous aurez une méthode **testPoint()** qui illustrera la création, l'affichage, la translation, … d'un Point. Pour chacune des classes à créer ou des fonctionnalités à ajouter vous aurez donc ce programme de test à compléter.
Pour chaque classe écrite et chaque méthode, vous devez **commenter** en utilisant la **syntaxe javadoc.**
Pour débuter vous pouvez **récupérer les classes *Figure, Point* et *test*** (http://calamar.univ-ag.fr/uag/ufrsen/coursenligne/egrandch/java/tp1.tar) qui vous servirons de base, vous les compléterez (méthodes, entête, attributs, …) au fur et à mesure ou utiliser les classes que vous avez généré lors du précédent TP.

1. Coder la classe ***Figure*** : faire remonter dans cette classe abstraite toutes les méthodes et tous les attributs communs aux figures, coder les méthodes qui peuvent l'être.
2. Coder la classe ***Point*** possédant comme attributs *les coordonnées du point* et en ajoutant les méthodes suivantes.
    a. Constructeur : ***public Point(double x, double y)***
    b. Constructeur : ***public Point(Point p)***
    c. Calcul de distance : ***public double distance(Point p)***
3. Coder la méthode ***testPoint()*** de la classe *test*
4. Coder la classe ***Segment*** possédant comme attributs les *deux points extrêmes* du segment et en ajoutant les méthodes suivantes.
    a. Constructeur : ***public Segment(Point p1, Point p2)***
    b. Calcul de longueur : ***public double getLongueur()***
    c. *Calcul du milieu du segment :* ***public Point getMilieu()***
5. Coder la méthode ***testSegment()*** de la classe *test*

6. Coder la classe ***Cercle*** possédant comme attributs un *centre* et un *rayon* et les méthodes suivantes.
    a. Constructeur : ***public Cercle(Point centre, double rayon)***
    b. Constructeur : ***public Cercle(Point p1, Point p2)***
7. Coder la méthode ***testCercle()*** de la classe *test*
8. Coder l'interface ***Nomme*** qui possède la méthode ***getNom()***
9. Coder la classe ***PointNomme*** héritant de *Point* et implémentant *Nomme*.
10. Coder la méthode ***testPointNomme()*** de la classe *test*
11. Coder la classe ***SegmentNomme*** héritant de *Segment* et implémentant *Nomme*.
12. Coder la méthode ***testSegmentNomme()*** de la classe *test*
13. Illustrer le polymorphisme en passant des *PointNomme* au lieu des *Point* dans le constructeur de *Segment*. Coder cela dans la méthode ***testPolymorphisme()*** de la classe *test*.
14. Coder la méthode *equals()* pour les classes *Point, Segment et Cercle* comme vu en TD
    a. public boolean equals(….)
    b. On rappelle que
        i. Deux points sont égaux si leurs coordonnées sont identiques
        ii. Deux segments sont égaux si les deux points extrêmes sont égaux quelque soit leur position respective.
        iii. Deux cercles sont égaux si ils ont même centre et même rayon.
15. Compléter les différentes méthodes de test afin de tester l'égalité.
16. Ajouter le clonage en profondeur aux classes *Point, Segment et Cercle* comme vu en TD (implémentation de *Cloneable* et codage de la méthode *clone()*).
17. Compléter les différentes méthodes de test afin de tester le clonage.
18. Coder la sérialisation pour les classes *Point, Segment et Cercle* comme vu en TD (implémentation de *Serializable*).
19. Coder la méthode ***testSerialization()*** dans la classe *test* qui teste la sérialisation.
20. Coder une méthode qui permet de tester le fonctionnement des structures de données (méthode ***testStructures()*** de la classe *test*) avec les figures. Construire par exemple une liste de *Figure* en y ajoutant des *Point, Cercle, PointNomme, …* puis en affichant la liste, en la translatant, …. Tester également la recherche de figure dans la liste.
21. Coder la classe Polygone à partir d'une liste de Points avec l'ensemble des méthodes développées pour les autres figures (clonage, égalité, …) et en ajoutant les méthodes suivantes
    a. ***public boolean add(Point p)*** : cette méthode ajoute un *Point* au *Polygone* sans faire de vrifications, elle renvoie *true* systématiquement. La valeur retournée est mise pour permettre éventuellement de dériver la classe et de coder une méthode *add()* qui n'autoriserait pas de doublons dans la liste des points ou alors qui n'autoriserait pas les polygones croisés. Dans ces cas, la méthode pourrait renvoyer *false.*
    b. Remarque : deux polygones sont égaux si il existe une correspondance entre les deux listes de points (les points ne sont pas forcément en correspondance directe : P1 avec P1, P2 avec P2, …).
22. Coder la méthode ***testPolygone()*** de la classe *test*
23. Vous devez remettre à votre enseignant de TP les fichiers suivants : ***Point.java, Segment.java, Cercle.java, Polygone.java, test.java.***